Exhibit A

Claim(s)	Phrase for Construction	eBay's Proposed Construction and Support	MasterObjects' Proposed Construction and Support
'639: 1, 13 '529: 1	asynchronous connection	**Intrinsic:** The '529 Patent, at Cols. 12:22-36. The '639 Patent, at Cols. 14:25-39. The '326 Patent, at Cols. 14:48-54. '529 Patent File History, Response to Office Action, dated 12/19/2005, at 12. U.S. Provisional App. No. 60/622,907 at ¶ 55. **Extrinsic:** Plaintiff's Opening Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 40 (N.D. Cal.). Yahoo!'s Responsive Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 42 (N.D. Cal.). MasterObjects's Reply to Yahoo!'s Responsive Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 44 (N.D. Cal.). Tentative Rulings and Questions Re Claim Construction, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 49 (N.D. Cal.).	**Intrinsic:** A system . . . that allows the client system to send via the network, and within a session between the client system and the server system, a lengthening string composed of a plurality of consecutively input characters . . . while asynchronously receiving consecutive responses from the server system as the characters are being input ('529 Claim 1). asynchronously returns, while the increasingly lengthening query string is being entered by the user at the input field at the client, increasingly relevant content to the client. ('639 Claim 1). The invention relates generally to client-server communication systems, and particularly to a session-based bi-directional multi-tier client-server asynchronous search and retrieval system. ('529, 1:15-20) Roughly described, the invention provides a session-based bi-directional multi-tier client-server asynchronous information database search and retrieval system for sending a character- by-character string of data to an intelligent server that can be configured to immediately analyze the lengthening string character-by-character and return to the client increasingly appropriate database information as the client sends the string. ('529, 8:25-33) The system is bi-directional and asynchronous, in that both the Client and the Server can initiate communications at any moment in time. The functionality of the system is such

		Transcript of the Deposition of Stefan van den Oord on September 26, 2012, continuing through September 28, 2012.	that it can run in parallel with the normal operation of clients. Tasks that clients execute on the system are non-blocking, and clients may resume normal operation while the system is performing those tasks. For example, a communication initiated by the Client may be a single character that is sent to the Server, that responds by returning appropriate data. An example of a communication initiated by the Server is updating the information provided to the client. Because the system is session-based it can keep track of database information that has been sent to the Client. As information changes in the database, the Server sends an updated version of that information to the Client. (‘529, 12:22-36) The present invention provides a system that allows clients or client applications to asynchronously retrieve database information from a remote server or server application. (‘529, 11:45-48) Finally, result sets include an identifier 'resultSetId'. Every time a Client Quester uses the protocol of the present invention to send something to the Server Quester that may result in a new QuestObjects Result Set, it includes a request identifier. This identifier is then copied in the resultSetId when the QuestObjects Result Set is sent to the Client Quester. In this way Client Questers know which request the QuestObjects Result Set belongs to. (This is important because the system is asynchronous and on occasions it may occur that a newer QuestObjects Result Set is sent to the client before an older one. (‘529, 23:27-38) The present invention can be implemented on any client and server system using any combination of operating systems and programming languages that support asynchronous network connections and preferably but not

			necessarily preemptive multitasking and multithreading. (‘529, 30:53-57) ***Prosecution History:*** Furthermore, since the system is *asynchronous*, both the client and the server can initiate communications at any moment in time. A net result of this is that, for example, as content information changes at the server, or as the server receives input from the client, the server can automatically match the extending query string, (for example, the lengthening or the shortening string), against its content, and can automatically send updated results to the client, without the client user having to click "submit". (‘529 PH, 12/19/2005 RCE, p. 12) The advantages of such a session-based protocol include that the server recognizes when subsequent requests originate at the same client. Thus, in responding to an input character the server receives from the client, the server can use the history of data that has already been sent to/from that client. Furthermore, since the system is *asynchronous*, both the client and the server can initiate communications at any moment in time. A net result of this is that, for example, as content information changes at the server, or as the server receives input from the client, the server can automatically match the extending query string, (for example, the lengthening or the shortening string), against its content, and can automatically send updated results to the client, without the client user having to click "submit". This is a very user-friendly means of searching complex server content and databases. By way of illustration, when used in a Web-site search application, since the server monitors each query as it is lengthened or shortened by one or more characters,

			automatically matches the focused query string against the content of the server system, and asynchronously returns increasingly relevant content information to the client object for immediate use by the client system, while a user of a client is typing a search input into the application the server can simultaneously (because of it asynchronous properties) communicate relevant information (i.e. feedback} to the client, so that the search options presented to the user can be immediately revised. In this manner, the application suggests to the user increasingly appropriate search options based both on their current input and on the present content of the server. ('529 PH, 12/19/2005 Amendment, p. 12) The advantages of the embodiment currently defined by Claim1 include that the server knows immediately when a search string character is entered by an end-user at the client, without the end-user having to, e.g., click "submit" each time. The server can then respond automatically as each character is being received at the client object, and as the query string is being lengthened. This immediate feedback may, for example, be in the form of providing preliminary search results, or suggesting a more appropriate search string. In one embodiment, the feedback can resemble a form of auto-complete or auto-suggestion, which is presented to the user as the user is actually typing (i.e., lengthening) a search string, but without the user having to click on a "submit" button. Matching content information is asynchronously returned to the user, while the user is still entering their query. ('529 PH, 5/4/2007 Response to Office Action, p. 13). Furthermore, as defined by Claim 1, when the server receives queries it *asynchronously* returns increasingly matching content information to the client object for immediate use by the client system. Unlike Zim, this

			feature allows data to be returned to the user as the user is actually entering a search string. (‘529 PH, 9/11/2006 Response to Office Action, p. 14) Claim 1 has been amended to more clearly define that the communication protocol enables an asynchronous connection *over* a *network* between a client system and a server system. The communication protocol allows the client system to send via the network, and within a session between the client system and the server system, a lengthening string composed of a plurality of consecutively input characters, *while receiving an asynchronous response from the server* as *the characters are being input.* The client object receives additional characters from the client software, and as consecutive characters are being received, transmits *via the network* to a server object at the server system a plurality of consecutive queries, within the same session, to retrieve content from the server system. (‘529 PH, 10/31/2007 Response to Office Action, p. 17) Applicant respectfully submits that, based on the above description, it appears that, in Payne, the client and the database it accesses must reside on the same physical device for proper operation, i.e. there does not appear to be any separation between the client and its database (or between, e.g. a client and a server). Claim 1 has been amended to more clearly define that, in the embodiment therein, the communication protocol enables an asynchronous connection *over* a *network* between a client system and a server system, and allows the client system to send via the network, and within a session between the client system and the server system, a lengthening string composed of a plurality of consecutively input characters, *while receiving an asynchronous response from the server* as *the characters are being input.* The client object receives additional characters from the client software, and as

			consecutive characters are being received, transmits, again *via the network* to a server object at the server system a plurality of consecutive queries, within the same session, to retrieve content from the server system. Applicant further respectfully submits that, based on the above description, it appears that, in Danielsen, the system may be considered asynchronous only within the context of distinguishing real-time data conferencing (i.e. synchronous conferencing) from non real-time data conferencing (i.e. asynchronous conferencing). Danielsen uses the term "asynchronous" to refer to the time aspect of the meeting, and further describes that the term asynchronous is meant that the participants do not have to be attending the session simultaneously. Claim 1 has been amended to clearly distinguish the use of this term, and to more clearly define that, in the embodiment therein, it is the communication protocol that enables an asynchronous connection over a network between a client system and a server system, and allows the client system to send via the network, and within a session between the client system and the server system, a lengthening string composed of a plurality of consecutively input characters, to query the server system for string-based content, *while receiving an asynchronous response from the server* as *the characters are being input.* (‘529 PH, 10/31/2007 Response to Office Action, p. 19) However, Applicant respectfully submits that Yano in these sections appears to disclose that, when the user identifies a desired document out of a pop-up menu through a mouse operation the search-engine-compatible interface unit notifies the URL (identification of the document) of the corresponding document-described on the list to the browser. As shown in the corresponding Figures 3 and 6, these steps (s306, s308, s609, s61 0) appear to be performed at the client. Only then does the browser access

			the database (Web server) with a document in the linked address stored therein according to the URL, and requests the database to transfer the source code of the corresponding document. As such, Applicant respectfully submits that Yano does not disclose that the server object asynchronously returns, *while the additional characters are being input and the string is being lengthened,* increasingly matching content information to the client. ('529 PH, 8/20/2008 Response to Office Action, p. 26) Gershman was cited in the Office Action as disclosing the use of a communications protocol that enables an asynchronous connection over a network between a client system and a server system. Applicant respectfully submits that Gershman appear to disclose an Asynchronous Messaging component that can provide an asynchronous message based communication between the client and the server, using standard Internet mail protocols, SMTP and POP3s. (Column 48, lines 9-18). However, while SMTP and POP provide a form of email-based communication between a client and a server, Applicant respectfully submits that neither Gershman (nor Yano) disclose or render obvious the specific feature of a communication protocol that allows the client system to send a lengthening string composed of a plurality of consecutively input characters, to query the server system for string-based content, while asynchronously receiving consecutive responses from the server as the characters are being input. Claim 1 has also been amended as shown above to more clearly provide this distinction. ('529 PH, 5/13/2009 Response to Office Action, p. 26) As such, Applicant respectfully submits that Yano does not appear to disclose or render obvious several features of Claim 1, including for example that the client object receives additional characters from the client software, and

			as *consecutive characters are being received, transmits via the network to* a *server object at the server system* a *plurality of consecutive queries,* within the session between the client system and the server system, to retrieve content from the server system, *wherein each consecutive query lengthens the string by the additional characters,* to form a lengthening string for retrieving matching content from the server system; and that the server object automatically uses the lengthening string to query and retrieve content information from the server system, and asynchronously returns, *while the additional characters are being input,* consecutive responses containing increasingly matching content information to the client. ('529 PH, 5/13/2009 Response to Office Action, p. 21) However, it was asserted that one of ordinary skill in the art would have found it obvious to implement or incorporate Hailpern's session between the client and server systems in Yano in order to return to the user a list of URL addresses based on the search for the partially specified URL address; that the use and advantage of asynchronously receiving consecutive responses is well known to one of ordinary skill in the art as evidenced by Chua; and that one of ordinary skill in the art would have found it obvious to incorporate or implement Chua's asynchronously returning responses in Yano's system providing the client with search results. Applicant respectfully traverses the foregoing assertion. As currently presented, Claim 1 recites a communication protocol that enables an asynchronous connection *over* a *network between* a *client system and* a *server system,* and allows the client system to send via the network, and within a session between the client system and the server system, a lengthening string composed of a plurality of consecutively input characters, to query the server system for stringbased

			content, *while asynchronously receiving consecutive responses from the server* as the characters are being input. Applicant respectfully submits that none of the cited references appears to disclose such a communication protocol that enables an asynchronous connection *over* a *network between* a *client system and* a *server system.* ('529 PH, 5/13/2009 Response to Office Action, pp. 22-23) As disclosed by Chua, the search system described therein appears to allow for easier addition of search engines, wherein each search engine registers with the search system through a wrapper, and wherein the wrapper provides communication between a search engine manager and the search engine. In Chua, when a query is initiated by a client, the query is apparently transmitted in series from the search engine manager to each of the several search engines, at which point the search engines can process the query in parallel with one another. The results from the search engines are not returned asynchronously to the client, but are instead returned to the search engine manager. As such, the client and the search engine manager do not appear to communicate or exchange results asynchronously; and particularly do not communicate asynchronously as new queries are being input at the client. Figures 5 and 6 of Chua also appear to confirm that the search client is not involved in the asynchronous receiving of results from the several search engines. Rather, it appears that only after a complete query is received from the client (e.g. step 530) is the query sent to each registered search engine; and only after all results are received at the search engine manager (e.g. step 660) are the results communicated back to the client. As such, Applicant respectfully submits that neither Yano, Hailpern, nor Chua appear to describe *an asynchronous session based connection between* a *client system and* a

			server system, or that the client can *asynchronously receive consecutive responses from the server* as *the characters are being input*, as recited by Claim 1. ('529 PH, 5/13/2009 Response to Office Action, p. 23) Applicant respectfully traverses this assertion. Although Chua appears to indicate that, when initiated by a client, "a query is transmitted to the search engines in series, the search engines execute the query in parallel, and the results are returned asynchronously to the client", Chua also appears to describe that the search engine manager calls each wrapper registered to handle queries for participating search engines, and that the wrappers may be called to execute their respective searches asynchronously in parallel. Figures 5 and 6 of Chua similarly appear to indicate that the client is not involved in the asynchronous receiving of results from the several search engines; rather, it appears that only after a complete query is received from the client (e.g. step 530) is the query passed to each registered search engine; and only after all of the responses are received at the search engine manager (e.g. step 660) are the results communicated back to the client. As such, it appears that the asynchronous feature referred to by Chua provides that a query, once received at the search engines, can be processed in parallel, and the search results of each search engine returned as those searches are completed (i.e. not necessarily in the order in which they were initiated). However, Applicant respectfully submits that neither Yano, Hailpern, nor Chua appear to describe an asynchronous session based connection between a client system and a server system in which the client can asynchronously receive consecutive responses from the server, as characters are being input, as recited by Claim 1. To more clearly recite the embodiment therein, Claim 1 has been amended to recite that the client object receives, as

			input, consecutive additional characters from the client software, and while each of the consecutive additional characters are being received as input, transmits via the network to a server object at the server system one or more corresponding consecutive queries, within the session between the client system and the server system, to retrieve content from the server system, wherein each of the consecutive queries lengthens the string by the additional characters, to form a lengthening string for retrieving matching content from the server system. Claim 1 has also been amended to recite that the server object, in response to receiving each of the consecutive queries that modify the lengthening string, automatically uses the modified lengthening string to query and retrieve content information from the server system that matches the modified lengthening string, and asynchronously returns, while the additional characters are being input and the consecutive queries are being transmitted and the lengthening string is being modified during the session, consecutive responses containing content information which increasingly matches the modified lengthening string, to the client object for immediate use by the client system. Applicant respectfully submits that these features are neither disclosed by, nor obvious in view of Yano, Hailpern and/or Chua. ('529 PH, 2/19/10 Reply to Office Action, p. 22) Asynchronously retrieving information from a remote server is described in at least paragraphs [0061], [0064], [0110] and [0141]. ('529 PH, 8/22/2011 Appeal Brief, p. 3) In Yano, the request to the search engine is done after all of the characters of the word are obtained. There is no server object that "asynchronously returns, while the additional characters are being input and the consecutive queries are being transmitted and the lengthening string is being modified during the session, consecutive responses

			containing content information which increasingly matches the modified lengthening string" as in claim 1 of the present application. (‘529 PH, 8/22/11 Appeal Brief, p. 28) **Extrinsic:** **Asynchronous** *adj*. Pertaining to being, or characteristic of something that is not dependent on timing. For example asynchronous communications can start and stop at any time instead of having to match the timing governed by a clock. Microsoft Computer Dictionary (5th ed. 2002). **Asynchronous communications** *n*. Computer-to-computer communications in which the sending and receiving computers do not rely on timing as a means of determining where transmissions begin and end. Microsoft Computer Dictionary (5th ed. 2002).
‘639: 1, 13 ‘529: 1, 44	communication protocol	**Intrinsic:** The ‘529 Patent, at Cols. 11:55-59; 12:48-53; 20:11-20. The ‘639 Patent, at Cols. 13:58-62. The ’326 Patent, at Cols. 8:27-42; 14:14-18; 24:47-54. U.S. Provisional App. No. 60/622,907 at ¶ 52. **Extrinsic:** Microsoft Computer Dictionary (5th ed. 2002) (Definition of “Communications protocol”). Plaintiff’s Opening Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-	**Intrinsic:** These Clients use a communication protocol 102 to send information, including but not limited to single characters, and to receive information, including but not limited to lists of strings and corresponding metadata. (‘529, 12:49-53) The invention includes a Server, that handles requests for information from clients, and a communication protocol that is optimized for sending single characters from a Client to the Server, and lists of strings from the Server to the Client. (‘529, 11:55-59) Unlike existing data-retrieval applications, server data can be accessed through a single standardized protocol that can be built into programming languages, user interface components or web components. (‘529 6:66-7:2, 9:42-45) The system's protocol is not restricted to sending single characters. In fact, Clients can also use the protocol to send

		2539-JSW, Dkt. No. 40 (N.D. Cal.). Yahoo!'s Responsive Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 42 (N.D. Cal.). MasterObjects's Reply to Yahoo!'s Responsive Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 44 (N.D. Cal.). Tentative Rulings and Questions Re Claim Construction, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 49 (N.D. Cal.). Transcript of the Deposition of Stefan van den Oord on September 26, 2012, continuing through September 28, 2012.	a string of characters. For example, when a user replaces the contents of an entry field with a new string, the Client may then send the entire string all at once to the Server, instead of character by character. ('529, 12:3-8) The protocol of the present invention is implemented in the Client Controller and the Server Controller **400.** ('529, 18:17-19) To support this, the protocol of the present invention provides a number of messages that allow the Client Quester to send just the changes to the input buffer, instead of sending the entire input buffer. These messages include but are not limited to: inputBufferAppend, inputBufferDeleteCharAt, inputBufferinsertCharAt, inputBufferSetCharAt, inputBufferSetLength, and inputBufferDelete. After thus updating the Server Quester's input buffer, the Client Quester activates the result retriever to wait for new results and process them (step612). ('529, 20:11-20) Finally, result sets include an identifier 'resultSetId'. Every time a Client Quester uses the protocol of the present invention to send something to the Server Quester that may result in a new QuestObjects Result Set, it includes a request identifier. ('529, 23:26-30) Another important entity in FIG. ***8A*** is QoController. QoController is the entity that implements the protocol of the present invention. ('529:25:17-19) FIG. **8B** displays the minimal entities every QuestObjects Client must have. Every client of the QuestObjects System at least has a Client Controller QoClientController. QoClientController is a subclass of QoController that implements the client side of the protocol of the invention.

			('529, 25:43-48) FIG. *8C* shows the server part of the embodiment of the present invention, and includes the QoServerQontroller, one of the subclasses of QoController. QoServerController implements the server-side part of the protocol of the present invention. ('529, 25:55-59) The protocol of the present invention can be implemented on top of networking standards such as TCP/IP. ('529, 30:65-68) **Prosecution History:** A communication protocol is described in at least paragraphs [0061], [0066], [0068] and Figure 2. ('529 PH, 8/22/11 Appeal Brief, p. 3) Applicant respectfully traverses the foregoing assertion. As currently presented, Claim 1 recites a communication protocol that enables an asynchronous connection *over* a *network between* a *client system and* a *server system*, and allows the client system to send via the network, and within a session between the client system and the server system, a lengthening string composed of a plurality of consecutively input characters, to query the server system for string based content, *while asynchronously receiving consecutive responses from the server* as the characters are being input. Applicant respectfully submits that none of the cited references appears to disclose such a communication protocol that enables an asynchronous connection *over* a *network between* a *client system and* a *server system.* ('529 PH, 5/13/2009 Reply to Office Action, p. 22-23) Gershman was cited in the Office Action as disclosing the use of a communications protocol that enables an

			asynchronous connection over a network between a client system and a server system. Applicant respectfully submits that Gershman appear to disclose an Asynchronous Messaging component that can provide an asynchronous message based communication between the client and the server, using standard Internet mail protocols, SMTP and POP3s. (Column 48, lines 9-18). However, while SMTP and POP provide a form of email-based communication between a client and a server, Applicant respectfully submits that neither Gershman (nor Yano) disclose or render obvious the specific feature of a communication protocol that allows the client system to send a lengthening string composed of a plurality of consecutively input characters, to query the server system for string-based content, while asynchronously receiving consecutive responses from the server as the characters are being input. Claim 1 has also been amended as shown above to more clearly provide this distinction. ('529 PH, 8/20/2008 Reply to Office Action, p. 26) Claim 1 has been amended to more clearly define that the communication protocol enables an asynchronous connection *over* a *network* between a client system and a server system. The communication protocol allows the client system to send via the network, and within a session between the client system and the server system, a lengthening string composed of a plurality of consecutively input characters, *while receiving an asynchronous response from the server* as *the characters are being input.* ('529 PH, 10/31/2007 Response to Office Action, p. 17) Claim 1 has been amended to more clearly define that, in the embodiment therein, the communication protocol enables an asynchronous connection *over* a *network* between a client system and a server system, and allows the

			client system to send via the network, and within a session between the client system and the server system, a lengthening string composed of a plurality of consecutively input characters, *while receiving an asynchronous response from the server* as *the characters are being input.* The client object receives additional characters from the client software, and as consecutive characters are being received, transmits again *via the network* to a server object at the server system a plurality of consecutive queries, within the same session, to retrieve content from the server system. (‘529 PH, 10/31/2007 Response to Office Action, p. 19) Applicant further respectfully submits that, based on the above description, it appears that, in Danielsen, the system may be considered asynchronous only within the context of distinguishing real-time data conferencing (i.e. synchronous conferencing) from non real-time data conferencing (i.e. asynchronous conferencing). Danielsen uses the term "asynchronous" to refer to the time aspect of the meeting, and further describes that the term asynchronous is meant that the participants do not have to be attending the session simultaneously. Claim 1 has been amended to clearly distinguish the use of this term, and to more clearly define that, in the embodiment therein, it is the communication protocol that enables an asynchronous connection over a network between a client system and a server system, and allows the client system to send via the network, and within a session between the client system and the server system, a lengthening string composed of a plurality of consecutively input characters, to query the server system for string-based content, *while receiving an asynchronous response from the server* as *the characters are being input.*) (‘529 PH, 10/31/2007 Response to Office Action, p. 19) As currently amended, the system comprises a

			communication protocol that provides an *asynchronous session-based connection* between a client and a server. As part of a session, the client can send a plurality of consecutively input strings to query the server for content. A client object transmits to a server object a plurality of consecutive queries, within the same session, to retrieve content from the server system. Each consecutive query either *lengthens or shortens the query string by one or more characters*, to form an *increasingly focused query string* for retrieving content from the server system. (For example, the query string can be lengthened by an additional character). A server object records, during the *same* session, each of the plurality of queries. As the search string is being lengthened or shortened, the server object *automatically matches the focused query string* against the content of the server system, and returns *increasingly relevant* content information to the client for immediate use by the client system. Applicant respectfully submits that these features are not disclosed by the cited references. The advantages of such a session-based protocol include that the server recognizes when subsequent requests originate at the same client. Thus, in responding to an input character the server receives from the client, the server can use the history of data that has already been sent to/from that client. ('529 PH, 12/19/2005 RCE, p. 12) **Extrinsic**: Communications protocol *n.* A set of rules or standards designed to enable computers to connect with one another and to exchange information with as little error as possible… MS Computer Dictionary, 5th Ed. (2002)
'529: 1, 44 '639: 1, 13	content-based cache	**Intrinsic:** The '529 Patent, at Cols. 9:17-20; 10:16-26; 15:9-	**Intrinsic:** Fig. 2

		13. The '639 Patent at Cols. 9:62-10:11; 10:12-22; 38:1-15. The '326 Patent at Cols. 5:3-8; 8:7-10; 38:58-39:3. U.S. Provisional App. No. 60/622,907 at ¶ 51. '529 Patent File History, Appeal Brief, dated 8/22/2011, at 25-26, 30. **Extrinsic:** Microsoft Computer Dictionary (5th ed. 2002) (Definition of "cache"). Plaintiff's Opening Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 40 (N.D. Cal.). Yahoo!'s Responsive Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 42 (N.D. Cal.). MasterObjects's Reply to Yahoo!'s Responsive Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 44 (N.D. Cal.). Tentative Rulings and Questions Re Claim Construction, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 49 (N.D. Cal.). Transcript of the Deposition of Stefan van den	Fig. 8A-2 Content-based Cache-A persistent store of Queries and corresponding Result Sets executed by a Content Engine for a specific Content Channel. ('529, 10:17-19) Content Channels may access a content-based cache **222** in which information that was previously retrieved from Content Engines is cached. ('529, 15:10-13) The present invention, however, stores and retrieves the auto-complete suggestions from databases on the server. Using the present invention, the suggestions generated by the server may, at the option of the application developer, be cached on the middle tier or on the client itself to maximize performance ('529, 6:49-55) Caching is an optimization as well. Caching is done by the QoResultsCache entry, to which the QuestObjects Controller has a reference. The QoResultsCache has a list of cached entries ('resultsCacheEntries'). The entry of the cache is modeled as QoResultsCacheEntry, an entity that has a list of QuestObjects Result Sets for combinations of query strings and qualifiers (as defined in QoQuery). ('529, 25:27-34) The present invention, however, stores and retrieves the auto-complete suggestions from databases on the server. Using the present invention, the suggestions generated by the server may, at the option of the application developer, be cached on the middle tier or on the client itself to maximize performance ('529, 8:41-46) In the detailed description below, an embodiment of the present invention is referred to as QuestObjects, and provides a system of managing client input, server queries,

		Oord on September 26, 2012, continuing through September 28, 2012.	server responses and client output. One specific type of data made available through the system from a single source (or syndicate of sources) is referred to as a QuestObjects Service. Other terms used to describe the QuestObjects system in detail can be found in the glossary below: (‘529, 9:47-56) The Server Controller may look up query 'a' in its Result Set cache, in which case it can return a previous Result Set to the Client without accessing the Service. (‘529, 18:52-56) After receiving a query, the Service executes it by accessing its Content Engine through the Content Access Module unless the Query Manager was able to lookup the same Query with a Result Set in the Content-based Cache. (‘529, 18:68-19:4) The Server Quester will stop the 'repeating' of query 'ab' and the Service will now execute 410 the new query 'abc' at the Content Engine, or retrieve it from the Content-based Cache. (‘529, 19:28-31) If the query is valid, the Server Quester checks if the server-side cache has the results for the Query String (step 709). If not, a new Query is sent to the Service (step 71 0). After that, the results are retrieved (either from cache or from the Service) and processed (step 711). (‘529, 20:63-68) The processing of query results is performed in a separate thread of execution. The process performed in this thread starts by obtaining the Result Set (step **714),** either from the server-side cache or from the Service depending on the result of the decision in step **709** (‘529, 21:5-10)

			End users of the present invention experience an unprecedented level of user-friendliness accessing information that is guaranteed to be up-to-date while being efficiently cached for speedy access as the number of simultaneous users grows. (‘529, 30:48-53) a content-based cache, at the server system, which stores previous queries and corresponding result sets previously executed by the system, and which includes within its result sets content or other information previously retrieved from the server system or one or more content sources in response to the previous queries. (‘529 Claim 1) ***Prosecution History:*** A content-based cache is described in at least paragraphs [0060], [0076], [0091], [0092], [0117], [0140], [0156] and Figures 2, 7A, 7B and 8A. (‘529 PH, 8/21/11 Appeal Brief, p. 3) **2. Content-Based Cache** Claim **1** includes the feature of: a content-based cache, at the server system, which stores previous queries and corresponding result sets previously executed by the system, and which includes within its result sets content or other information previously retrieved from the server or one or more content sources in response to the previous queries. Pages 4-5 of the Office Action of December 28, 2010 state that Curtis shows this feature. Curtis does describe storing URLs and information related to the URLs. However, Curtis and the other references do not disclose or make obvious the claimed functionality of the cache of claim **1.** Claim **1** states that the cache stores "previous queries" and "corresponding result sets" (or in some other claims "other information previously retrieved from the server").

			Such information is not stored in Curtis; Curtis only stores URLs and information related to the URL in an index file. It is not clear to the Applicant whether Curtis shows a cache at all. The index files of Curtis are used to store URL information for a search, but do not apparently cache responses. ('529 PH, 8/22/11 Appeal Brief, p. 30) To more clearly recite the embodiment therein, Claim 1 has been amended to further recite that, in accordance with the embodiment therein, the system includes a content-based cache, at the server system, which stores previous queries and corresponding result sets, previously executed by the system, and which includes within its result sets content or other information previously retrieved from the server or one or more content sources in response to the previous queries; ('529 PH, 9/30/2010 Response to Office Action, p. 22)
'529: 1, 44 '639: 1, 13	query and result cache	**Intrinsic:** The '529 Patent, at Cols. 9:17-20; 10:16-26; 15:9-13. The '639 Patent at Cols. 9:62-10:11; 10:12-22; 38:1-15. The '326 Patent at Cols. 5:3-8; 8:7-10; 38:58-39:3. U.S. Provisional App. No. 60/622,907 at ¶ 51. '529 Patent File History, Appeal Brief, dated 8/22/2011, at 25-26, 30. **Extrinsic:**	**Intrinsic:** '639 Claims 1, 13: A query and result cache that stores one or more query strings previously communicated from the clients, or content results previously returned from the server.

		Microsoft Computer Dictionary (5th ed. 2002) (Definition of "cache"). Plaintiff's Opening Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 40 (N.D. Cal.). Yahoo!'s Responsive Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 42 (N.D. Cal.). MasterObjects's Reply to Yahoo!'s Responsive Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 44 (N.D. Cal.). Tentative Rulings and Questions Re Claim Construction, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 49 (N.D. Cal.). Transcript of the Deposition of Stefan van den Oord on September 26, 2012, continuing through September 28, 2012.	
'529: 1	content source(s)	**Intrinsic:** The '529 Patent, at Cols. 8:55-62; 10:24-26; 13:4-15; 15:21-29; 30:24-33. The '639 Patent, at Cols. 5:25-31; 10:12-31. The '326 Patent, at Cols. 10:11-32. U.S. Provisional App. No. 60/622,907 at ¶ 51. **Extrinsic:** Plaintiff's Opening Claim Construction Brief,	**Intrinsic:** A session-based client-server asynchronous information search and retrieval system for sending character-by-character or multi-character strings of data to an intelligent server, that can be configured to immediately analyze the lengthening string and return to the client increasingly appropriate search information. ('639 Patent at Abstract) the system offers a standardized way to access server data that allows immediate user-friendly data feedback based on user input. ('639 Patent at 5:17-19)

		MasterObjects, Inc. v. Yahoo! Inc., 3:11-CV-2539-JSW, Dkt. No. 40 (N.D. Cal.). Yahoo!'s Responsive Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 42 (N.D. Cal.). MasterObjects's Reply to Yahoo!'s Responsive Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 44 (N.D. Cal.). Tentative Rulings and Questions Re Claim Construction, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 49 (N.D. Cal.). Transcript of the Deposition of Stefan van den Oord on September 26, 2012, continuing through September 28, 2012.	The system can also be used to simply and quickly retrieve up-to-date information from any string-based content source. Strings can be linked to metadata allowing user interface components to display corresponding information such as, for example, the meaning of dictionary words, the description of encyclopedia entries or pictures corresponding to a list of names. ('639 Patent at 5:25-30) Unlike existing data-retrieval applications, server data can be accessed through a single standardized protocol that can be built into programming languages, user interface components or web components. The system can be integrated into and combined with existing applications that access server data. Using content access modules, the system can access any type of content on any server. ('639 Patent at 6:11-17) a communication protocol which enables an asynchronous connection between the clients and the server, and allows each client to communicate, under control of a user and as part of a session, a plurality of consecutive query strings to query the server for content. ('639 Patent at Claim 1) a query and result cache that stores one or more query strings previously communicated from the clients, or content results previously returned from the server; wherein each of the clients provides an input field, which allows the user to enter as an input a query comprised of a plurality of consecutive query strings, and wherein the client transmits to the server within the session a plurality of queries to retrieve content from the server matching or related to the plurality of consecutive query strings, wherein each of the plurality of queries form an increasingly lengthening query string for retrieving content from the server. ('639 Patent at Claim 1)

			in response to receiving each of one or more additional characters in the increasingly lengthening query string as they are being entered at the input field, automatically matches the increasingly lengthening query string initially by matching the query string against the content of the query and result cache, and subsequently by matching the query string against other content available to the server, and asynchronously returns, while the increasingly lengthening query string is being entered by the user at the input field at the client, increasingly relevant content to the client. ('639 Patent at Claim 1) The present invention can also be used to simply and quickly retrieve up-to-date information from any string-based content source. Strings can be linked to metadata allowing user interface components to display corresponding information such as, for example, the meaning of dictionary words, the description of encyclopedia entries or pictures corresponding to a list of names. ('529 Patent at 6:12-18) Roughly described, the invention provides a session-based bi-directional multi-tier client-server asynchronous information database search and retrieval system for sending a character-by-character string of data to an intelligent server that can be configured to immediately analyze the lengthening string character-by-character and return to the client increasingly appropriate database information as the client sends the string. ('529 Patent at 8:25-33) The present invention is useful for an extremely wide variety of applications. It offers a standardized way to access server data that allows immediate user-friendly data feedback based on user input. ('529 Patent at 8:47-50)

			Unlike existing data-retrieval applications, server data can be accessed through a single standardized protocol that can be built into programming languages, user interface components or web components. The present invention can be integrated into, and combined with, existing applications that access server data. Using Content Access Modules, the present invention can access any type of content on any server. (‘529 Patent at 9:42-48) a communication protocol that enables an asynchronous connection over a network between a client system and a server system, and allows the client system to send via the network, and within a session between the client system and the server system, a lengthening string composed of a plurality of consecutively input characters, to query the server system for string-based content, while asynchronously receiving consecutive responses from the server system as the characters are being input. (‘529 Patent at Claim 1) a content-based cache, at the server system, which stores previous queries and corresponding result sets previously executed by the system, and which includes within its result sets content or other information previously retrieved from the server system or one or more content sources in response to the previous queries. (‘529 Patent at Claim 1) in response to receiving each of the corresponding consecutive queries that modify the lengthening string, automatically uses the lengthening string to query and retrieve content information from the content-based cache at the server system or from the one or more content sources that matches the lengthening string. (‘529 Patent at Claim 1)
‘639: 1, 13	session	**Intrinsic:**	**Intrinsic:**

'529: 1, 44		The '529 Patent, at Cols. 2:38-52; 12:9-36; 25:64-26:17; 27:2-8. The '639 Patent at Cols. 8:7-11; 14:12-17; 24:41-45; 25:42-56; 26:42-47; 37:20-22. The '326 Patent at Cols. 8:3-6; 8:59-63; 24:64-25:5; 26:4-10; 27:3-11; 38:5-6 U.S. Provisional App. No. 60/622,907 at ¶¶ 54, 55, 98, 119, 185, 190, 191, 193. '850 Provisional App. at 16 '529 Patent File History, Response to Office Action, dated 5/4/2007, at 15. '529 Patent File History, Response to Office Action, dated 12/21/2005, at 11-13. '529 Patent File History, Response to Office Action, dated 4/11/2005, at 11-13. **Extrinsic:** Plaintiff's Opening Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 40 (N.D. Cal.). Yahoo!'s Responsive Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 42 (N.D. Cal.). MasterObjects's Reply to Yahoo!'s Responsive Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 44	**FIELD OF THE INVENTION** The invention relates generally to client-server communication systems, and particularly to a session-based bi-directional multi-tier client-server asynchronous search and retrieval system. ('529, 1:15-20) Unfortunately, the rise of Internet commerce has also given rise to some of the disadvantages associated with mainframe technology. Most Internet connections that present data to the user or client process use the Hyper Text Transfer Protocol (HTTP) which is inherently "session-less." This means that, for example, there is no totally reliable way for the server to automatically update the client display once the server data change. It also means that the server only checks the validity of the client or user input after the user sends back or submits an entire input form. This apparent disadvantage has also played an important role in the success of the Internet: because HTTP connections are session-less, they require much less processing power and much less memory on the server while the user is busy entering data. ('529, 2:38-52) What is needed is a mechanism that addresses these issues that allows a client-server system to retain some element of a session-based system, with its increase in performance, while at the same time offering a secure communication mechanism that requires little, if any, local storage of data. ('529, 3:19-24) Roughly described, the invention provides a session-based bi-directional multi-tier client-server asynchronous information database search and retrieval system for sending a character-by-character string of data to an intelligent server that can be configured to immediately analyze the lengthening string character-by-character and

		(N.D. Cal.). Tentative Rulings and Questions Re Claim Construction, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 49 (N.D. Cal.). Transcript of the Deposition of Stefan van den Oord on September 26, 2012, continuing through September 28, 2012	return to the client increasingly appropriate database information as the client sends the string. ('529, 8:25-34) In accordance with one embodiment of the invention the system is *session-based, in that the server knows or recognizes when subsequent requests originate at the same Client*. Thus, in responding to a character the Server receives from a Client it can use the history of data that has been sent to and from the current user. In one embodiment, the system stores user preferences with each Service, so that they are always available to the Client, (i.e., they are independent of the physical location of the client). Furthermore, client authentication and a billing system based on actual data and content use by Clients are supported. For faster response, the Server may predict input from the Client based on statistics and/or algorithms. ('529, 12:9-21) The system is bi-directional and asynchronous, in that both the Client and the Server can initiate communications at any moment in time. The functionality of the system is such that it can run in parallel with the normal operation of clients. Tasks that clients execute on the system are non-blocking, and clients may resume normal operation while the system is performing those tasks. For example, a communication initiated by the Client may be a single character that is sent to the Server, that responds by returning appropriate data. An example of a communication initiated by the Server is updating the information provided to the client. Because the system is session-based it can keep track of database information that has been sent to the Client. As information changes in the database, the Server sends an updated version of that information to the Client. ('529, 12:22-36) FIG. 2 displays just one session and one content Service. In

			an actual implementation there may be multiple concurrently active sessions, and there may be more than one content Service that Clients can use. (‘529, 12:65-13:1) Client Questers are managed by a Questlet, which create and destroy Questers they need. In a similar fashion, Server Questers are managed by a Session **207**. When a Client Quester is created, it registers itself with the Client Controller. The Client controller forwards this registration information as a message to the Session using the Server Controller. The Session then checks if the Persistent Quester Store **210** contains a stored Quester belonging to the current user matching the requested Service and Query Qualifier. If such a Quester exists, it is restored from the Persistent Quester Store and used as the peer of the Client Quester. Otherwise, the Session creates a new Server Quester to be used as the Client Quester's peer. (‘529, 13:55-67) The QuestObjects System is session-based. ***This means that clients that use the system are assigned to a session,*** modeled by the QoSession entity. Every session has a unique identifier, the ‘sessionid’. The QoSession entity maintains a list of Server Questers that are active in the session (stored in the ‘serverQuesters’ property). Furthermore, it has a reference to the Server Controller through which a QuestObjects Client is using the session. (‘529, 25:65-26:6) QoServerQuester is the server-side subclass of QoQuester. It includes a reference to the session it is being used in (the ‘session’ property). Furthermore, when the QuestObjects Service that the Quester uses has a Query Validator, QoServerQuester has (a reference to) a copy of that Query Validator, so that query strings can be validated before they are actually sent to the QuestObjects Service. The

			QoPersistentQuesterStore is an entity that is able to store a user's session and to restore it at some other time, even when the session would normally have expired or even when the same user is connecting from a different client machine." ('529, 26:7-17) QuestObjects Servers communicate with QuestObjects Services through the QoServiceSession. The QoServiceSession has a static reference to the QuestObjects Service it belongs to, as well as a static array of IP addresses of QuestObjects Servers that are allowed to connect to the QuestObjects Service. In some versions of the QoServiceSession the array of IP addresses can be replaced by a list of addresses and netmasks, or by IP address ranges. Every instance of QoServiceSession has the IP address of the server that is using the session ('serverAddress'), a connection Timeout indicating the maximum period of idle time before the Service Session is automatically ended, and a serviceSessionId that can be used to refer to the Service Session. ('529, 26:64-27:8) **Prosecution History:** Spaey discloses a system and method to synchronously access structured data using the Internet platform (browser) to access, search, filter and sort remotely stored data. The system includes three main parts: an interface, a communication protocol, and an intelligent server. The intelligent server's function is to provide the information the user requests via the interface by formatting the interface request for the database and formatting the results from the database for the interface. (Abstract) ***As further described therein, the server is sessionless, i.e. no information about past requests is required to process the current request.*** This means several servers can process incoming requests in parallel. (Paragraph [0087]).

			Notwithstanding the comments provided above with respect to the effective prior art date of the Spaey reference, Applicant further respectfully submits that Spaey does not appear to disclose the features of the present invention. In the Office Action, it was submitted that Spaey discloses a communication protocol that enables an asynchronous session. However, as described above it appears that in Spaey it may advantageous for the server to be ***sessionless, so that no information about past requests is required to process the current requests,*** and so that several servers can process incoming requests in parallel. ('529 PH, 12/4/2008 Response to Office Action, p. 15) These protocols are inherently "session-less", i.e. no session is maintained between the client and the server. ('529 PH, 4/11/2005 Response to Office Action, p. 11) The present invention provides a method of providing a more reliable and functional Internet -based application. Unlike the traditional HTTP protocol, a session is maintained between the client and the server using an asynchronous session-based communications protocol. Since a session is maintained between the client and the server, the server can immediately check the validity of the client or user input against the server content as the user is entering a portion of a search string. Unlike HTTP or another session-less protocols, the server does not have to wait until the user submits an entire input form. ('529 PH, 4/11/2005 Response to Office Action, p. 12) The manner in which the client system and the server system are connected by an asynchronous session-based connection has been more clearly defined to distinguish a session-based connection from the general concept of a "session" between two or more parties. ('529 PH,

			12/19/2005 Amendment, p. 11) The advantages of such a session-based protocol include that the server recognizes when subsequent requests originate at the same client. Thus, in responding to an input character the server receives from the client, the server can use the history of data that has already been sent to/from that client. Furthermore, since the system is *asynchronous*, both the client and the server can initiate communications at any moment in time. A net result of this is that, for example, as content information changes at the server, or as the server receives input from the client, the server can automatically match the extending query string, (for example, the lengthening or the shortening string), against its content, and can automatically send updated results to the client, without the client user having to click "submit". This is a very user-friendly means of searching complex server content and databases. (‘529 PH, 12/19/2005 Amendment, p. 12) The present invention provides a method of providing a more reliable and functional Internet -based application. Unlike the traditional HTTP protocol, a session is maintained between the client and the server using an asynchronous session-based communications protocol. Since a session is maintained between the client and the server, the server can immediately check the validity of the client or user input against the server content as the user is entering a portion of a search string. Unlike HTTP or another session-less protocols, the server does not have to wait until the user submits an entire input form. (‘529 PH, 4/11/2005 Reply to Office Action, p. 12) **Extrinsic:**

			session *n.* **1.** The time during which a program is running. In most interactive programs, a session is the time during which the program accepts input and processes information. **2.** In communications, the time during which two computers maintain a connection. **3.** A specific protocol layer in the ISO/OSI reference model that manages communication between remote users or processes. Microsoft Computer Dictionary (5th ed. 2002).
'529: 1	server object	**Intrinsic:** The '529 Patent, at Cols. 8:34-43; 12:9-11; 10:49-58; 13:21-33; 30:65–31:4. **Extrinsic:** Microsoft Computer Dictionary (5th ed. 2002) (Definition of "object-oriented programming"). Newton's Telecom Dictionary (17th ed. 2001) (Definition of "Object Oriented Programming") Plaintiff's Opening Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 40 (N.D. Cal.). Yahoo!'s Responsive Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 42 (N.D. Cal.). MasterObjects's Reply to Yahoo!'s Responsive Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 44 (N.D. Cal.). Tentative Rulings and Questions Re Claim Construction, *MasterObjects, Inc. v. Yahoo! Inc.*,	**Intrinsic:** The preceding detailed description illustrates software objects and methods of a system implementing the present invention. By providing a simple and standardized interface between Client components and any number of Content Engines that accept string-based queries, the present invention gives content publishers, web publishers and software developers an attractive way to offer unprecedented interactive, speedy, up-to-date and controlled access to content without the need to write an access mechanism for each content source. ('529, 30:24-33) Object-oriented languages and even non-object-oriented (database) systems have used component technologies to implement technical functionality. The NeXT step operating system from NeXT Computer, Inc. (which was later acquired by Apple Computer, Inc. and evolved into the Mac operating system Mac OS X) had an object-oriented architecture from its original beginnings, that allowed software developers to create applications based on predefined, well-tested and reliable components. Components could be 'passive' user interface elements (such as entry fields, scroll areas, tab panes etc.) used in application windows. But components could also be active and show dynamic data (such as a component displaying a clock, world map with highlight of daylight and night, ticker tape showing stock symbols, graphs showing

		3:11-CV-2539-JSW, Dkt. No. 49 (N.D. Cal.). Transcript of the Deposition of Stefan van den Oord on September 26, 2012, continuing through September 28, 2012.	computer system activity, etc.). The NeXT operating system used object frameworks in the Objective C language to achieve its high level of abstraction which is needed for components to work well. Later, Sun Microsystems, Inc. developed the Java language specification in part to achieve the same goal of interoperability. To date, Java has probably been the most successful 'open' (operating system independent) language used to build software components. It is even used on certain web sites that allow 'Java applets' on the user's Internet browser to continuously show up-to-date information on the client system. WebObjects, an object-oriented technology developed by Apple Computer, Inc. is an Internet application server with related development tools, which was first developed by NeXT Computer, Inc. WebObjects uses object oriented frameworks that allow distribution of application logic between server and client. Clients can be HTMLbased, but can also be Java applets. WebObjects uses proprietary technology that automatically synchronizes application objects between client and server. The layer that synchronizes data objects between the client and the server is called the 'Enterprise Object Distribution' (EODistribution), part of Apple's Enterprise Objects Framework (EOF), and is transparent to the client software components and the server software components. (‘529, 4:22-60). Quester-An intelligent non-visual object contained by an Active Component that links a QuestObjects StringList to an input buffer. Questers exist on both the QuestObjectsClient and the QuestObjects Server and can be specifically referred to as Client Quester and Server Quester. Questers communicate with each other through a QuestObjects Controller. (‘529, 10:49-55).

			Questers are objects that tie a QuestObjects input buffer (containing input from the Client) to a QuestObjects Result Set returned from a QuestObjects Server. Questers exist on both the Client and Server, in which case they are referred to as a Client Quester and a Server Quester, respectively. (‘529, 13:25-30). Requests for information are Query objects that are sent to and interpreted by a specific Service. Query objects contain at least a string used by the Service as a criterion for information to be retrieved, in addition to a specification of row numbers to be returned to the Client. (‘529, 14:12-17) By encapsulating a Quester into an object supplied with a programming language, a QuestObjects Service can be made available to its developers. (‘529, 18:1-3) Queries are executed by QoQueryExecutors. A Query Executor has a reference to the Service Session in which the Query is executed, it has a reference to the Query itself, and it also has a reference to the Server Quester that has the Query executed. This reference may be a remote object when Corba is being used, for example. If some proprietary protocol is used, it may just be the unique identifier of the Server Quester. (‘529, 29:10-16) ***Prosecution History:*** Server objects, such as Server Questers, are described in at least paragraphs [0067], [0069], [0071], [0109], [0111] and [0131] and Figures 2, 7 A, 8A and 8B. Further, paragraph [0120] states that "Figure 8C shows the server part of the embodiment of the present invention, and includes the QoServerController, one of the subclasses of QoController. QoServerController implements the server-

			side part of the protocol of the present invention." The QoServerController is also a server object. The QoServerController is described in at least paragraphs [0117], [0120] and [0121] and Figures 2, 4, 8A and 8B. ('529 PH, 8/22/2011 Appeal Brief, p. 4) **Extrinsic:** object *n.* **1.** Short for object code (machine-readable code). **2.** In object-oriented programming, a variable comprising both routines and data that is treated as a discrete entity. (Microsoft Computer Dictionary, 5th Ed. (2002)
'529: 1, 9	client object	**Intrinsic:** The '529 Patent, at Cols. 8:34-43; 12:9-11; 10:49-58; 13:21-33; 30:65–31:4. **Extrinsic:** Microsoft Computer Dictionary (5th ed. 2002) (Definition of "object-oriented programming"). Newton's Telecom Dictionary (17th ed. 2001) (Definition of "Object Oriented Programming") Plaintiff's Opening Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 40 (N.D. Cal.). Yahoo!'s Responsive Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 42 (N.D. Cal.). MasterObjects's Reply to Yahoo!'s Responsive Claim Construction Brief, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 44	**Intrinsic:** The preceding detailed description illustrates software objects and methods of a system implementing the present invention. By providing a simple and standardized interface between Client components and any number of Content Engines that accept string-based queries, the present invention gives content publishers, web publishers and software developers an attractive way to offer unprecedented interactive, speedy, up-to-date and controlled access to content without the need to write an access mechanism for each content source. ('529, 30:24-33) Object-oriented languages and even non-object-oriented (database) systems have used component technologies to implement technical functionality. The NeXT step operating system from NeXT Computer, Inc. (which was later acquired by Apple Computer, Inc. and evolved into the Mac operating system Mac OS X) had an object-oriented architecture from its original beginnings, that allowed software developers to create applications based on predefined, well-tested and reliable components. Components could be 'passive' user interface elements (such as entry fields, scroll areas, tab panes etc.) used in

		(N.D. Cal.). Tentative Rulings and Questions Re Claim Construction, *MasterObjects, Inc. v. Yahoo! Inc.*, 3:11-CV-2539-JSW, Dkt. No. 49 (N.D. Cal.). Transcript of the Deposition of Stefan van den Oord on September 26, 2012, continuing through September 28, 2012.	application windows. But components could also be active and show dynamic data (such as a component displaying a clock, world map with highlight of daylight and night, ticker tape showing stock symbols, graphs showing computer system activity, etc.). The NeXT operating system used object frameworks in the Objective C language to achieve its high level of abstraction which is needed for components to work well. Later, Sun Microsystems, Inc. developed the Java language specification in part to achieve the same goal of interoperability. To date, Java has probably been the most successful 'open' (operating system independent) language used to build software components. It is even used on certain web sites that allow 'Java applets' on the user's Internet browser to continuously show up-to-date information on the client system. WebObjects, an object-oriented technology developed by Apple Computer, Inc. is an Internet application server with related development tools, which was first developed by NeXT Computer, Inc. WebObjects uses object oriented frameworks that allow distribution of application logic between server and client. Clients can be HTML-based, but can also be Java applets. WebObjects uses proprietary technology that automatically synchronizes application objects between client and server. The layer that synchronizes data objects between the client and the server is called the 'Enterprise Object Distribution' (EODistribution), part of Apple's Enterprise Objects Framework (EOF), and is transparent to the client software components and the server software components. (‘529, 4:22-60). Quester-An intelligent non-visual object contained by an Active Component that links a QuestObjects StringList to an input buffer. Questers exist on both the QuestObjects Client and the QuestObjects Server and can

			be specifically referred to as Client Quester and Server Quester. Questers communicate with each other through a QuestObjects Controller. ('529, 10:49-55). Questers are objects that tie a QuestObjects input buffer (containing input from the Client) to a QuestObjects Result Set returned from a QuestObjects Server. Questers exist on both the Client and Server, in which case they are referred to as a Client Quester and a Server Quester, respectively. ('529, 13:25-30). By encapsulating a Quester into an object supplied with a programming language, a QuestObjects Service can be made available to its developers. ('529, 18:1-3) Queries are executed by QoQueryExecutors. A Query Executor has a reference to the Service Session in which the Query is executed, it has a reference to the Query itself, and it also has a reference to the Server Quester that has the Query executed. This reference may be a remote object when Corba is being used, for example. If some proprietary protocol is used, it may just be the unique identifier of the Server Quester. ('529, 29:10-16) **Prosecution History:** Client objects, such as Client Questers, are described in at least paragraphs [0067], [0069], [0093] and Figures 2, 4, 6A, 6B, 8A and 8B. ('529 PH, 8/22/2011 Appeal Brief, p. 4) **Extrinsic:** object *n.* **1.** Short for object code (machine-readable code). **2.** In object-oriented programming, a variable comprising both routines and data that is treated as a discrete entity.

			(Microsoft Computer Dictionary, 5th Ed. (2002)